Chapter 9

Software and computing

9.1 Overview

The task of the MINOS offline software is relatively simple compared to that of the large collider experiments. The MINOS detector is monolithic, a simple repetition of scintillator and steel planes, whereas collider experiments contain a multiplicity of different and complicated detector elements. As discussed in Chapter 6, the data rates in the far detector are very low and even those in the near detector are small compared with a pp or $p\bar{p}$ collider. Thus writing the offline analysis code is not expected to be a major load and the processing and data storage requirements are modest. This chapter discusses our plans to provide the effort and resources that will be needed both to process our data and to provide the basis for the subsequent physics analysis.

The functional requirements of the offline processing software are four-fold:

1. Generating realistic Monte Carlo events.
2. Finding the hits associated with events, both real data and Monte Carlo.
3. Separating the hits associated with a muon and fitting its momentum and direction through the magnetic field.
4. Analyzing the hadron/electron shower at the vertex for flavor content, energy and direction.

A considerable fraction of the code that would be required for a final system already exists and has been used in the definition of the physics capabilities of MINOS and the design of the MINOS detectors. It is written in Fortran-77 and runs under UNIX. As described in Section 9.2, it uses the ADAMO system to define the data structures and GEANT3 together with the Soudan 2 neutrino generation routines for the Monte Carlo simulation. However work is required to improve and extend the current code. We are confident that this system will provide a well-engineered and user-friendly offline software system. An estimate of the effort needed is given in Section 9.2.

However the MINOS collaboration is becoming increasingly concerned about the future of Fortran, and in particular about future support for the tools (such as CERNLIB, ADAMO

and ZEBRA) which will be vital for the maintenance of a Fortran system over the lifetime of MINOS. In recognition of this concern, and of the potential advantages offered by recent progress in programming techniques, the collaboration is investigating the use of a new C++ Object Oriented offline program system[1]. Section 9.3 describes the advantages of such a system and the effort and costs involved in its production.

In Section 9.4 we discuss the event rates expected and estimate the computing power and data storage required to deal with MINOS data and Monte Carlo events. We find that the requirements are modest. At the far detector a farm of ten 300 MHz processors, 50 Gigabytes of disk storage and a small permanent storage facility compatible with the Fermilab central store is needed. About 700 Gigabytes of data per year will be produced. At the near detector a farm of 20 processors for data and 6 for Monte Carlo together with 1.3 Terabytes per year of data storage is required. We expect to use the Fermilab central facilities to provide the near detector requirements.

Our data processing model, data distribution scheme and plans for physics analysis are detailed in Section 9.5.

The overall status the MINOS offline software system is summarized in Section 9.6

9.2 The current Fortran analysis code

The current MINOS software[2] is based on Fortran-77 code supported by a variety of non-commercial libraries such as CERNLIB. Only minimal deviations from the standard were permitted; the allowed extensions are commonly supported features such as the use of long names and the `#include` facility for common block synchronization. The Fortran-77 approach allowed the MINOS software group to proceed without the learning curve that an Object Oriented (OO) model would have entailed – MINOS collaborators could contribute by taking advantage of their prior knowledge and familiarity with packaged software.

The major addition to the traditional HEP software tools used by experiments at Fermilab was the inclusion of ADAMO[3] as the interface to the data structures. The ADAMO package is a CERN/DESY-supplied set of routines for bridging the gap between a ZEBRA/BOS memory manager and a more OO-oriented model. This package has also been used by the Aleph, Hermes, Zeus and Selex experiments. ADAMO completely hides the complexity of the ZEBRA memory manager and substitutes a unified access to the structures which provides more security against data corruption. In addition it provides more portability. Event files contain an embedded representation of the data model at the time it was created; this allows the data structures to evolve as the understanding of our needs change, while retaining the ability to read previously generated files without significant user intervention. The files themselves are an ADAMO structure overlaying a ZEBRA machine-independent format; this allows the event files to be exchanged between platforms. Event generation and analysis has occurred on SGI, Sun, HP, IBM-AIX, and DEC OSF/1 machines with essentially no machine-dependent code written by the MINOS software group.

The conceptual model of ADAMO represents the data in tabular form. Columns represent attributes (e.g., volume identifiers, or components of a 4-vector), while each row represents an individual object. Relationship links allow connections between different objects of the same or different types. Intrinsic support routines furnish indexing (sorting and selection)

along different attributes or combinations of attributes. These sorted lists are automatically maintained by the ADAMO system and are handled without excessive overhead. They are then available to all routines accessing the data structure. Using the indexing capabilities avoids the need for much of the code users would write to perform looping and sorting and which is often error prone and time consuming. Data integrity checks are also a standard feature of ADAMO.

Long term prospects of this approach are uncertain to some degree. The ADAMO package, while stable and without known bugs, is receiving only minimal support – the authors have been pressed into service on other projects. The ZEBRA and GEANT3 packages are due to have CERN support dropped in the forseeable future. The source code for all these packages is available, but modifications to support the idiosyncrasy of new platforms may prove to be difficult. However we expect a substantial user community to be committed to these packages throughout the lifetime of MINOS.

9.2.1 Beam simulation

Some of the neutrino oscillation tests, such as the ratio of ratios of charged to neutral current events in the far and near detectors, are relatively insensitive to details of the beam and beam simulation. Others, however, such as the charged current total energy spectrum test, may be systematics limited by knowledge of the beam. Hence beam simulation and comparison to data will be an important and time consuming task. The most detailed feedback to the beam simulation will come from monitoring the charged current event rate in the near detector, as a function of event energy, event vertex radius, and event time (since the magnetic fields in the pulsed horn system vary over the spill). Comparisons will also be needed with the beam muon monitoring system. Understanding the hadron production model in the target is especially crucial to a good understanding of the beam, and will have to be much more developed than was needed for beam design.

Three particle physics Monte Carlo programs have been used to predict neutrino fluxes in beam studies so far. `GNuMI`, `NUADA`, and `PBEAM` ν beam simulation packages trade off speed versus range of effects that are included, as shown in Table 9.1. Being essentially independently developed, they also serve as cross-checks of the calculations.

`NUADA`, originally written by Wilber Venus at CERN and modified and extended by David C. Carey at Fermilab[4], generates a matrix of production angles and momenta for $\pi^{\pm}$ and $K^{\pm}$ at the target, and tracks this "mesh" through the focusing system. At each step along each track, it integrates a neutrino flux at the detector which combines the production probability for that angle and momentum, the decay probability for that track, and the acceptance of the detector. Thus it is actually a calculation rather than a Monte Carlo. Continuing care is required to ensure that the granularity of the mesh is fine enough.

`PBEAM`, written by Noel Stanton at Kansas State University and with weighting methods incorporated by Wesley Smart at Fermilab, generates $\pi^{\pm}$, $K^{\pm}$, and K^0 in a Monte Carlo fashion, and tracks them through the focusing system. Absorption of hadrons in the horns is taken into account, but secondaries are not generated. Each hadron is then decayed at one position. `PBEAM` contains the option of generating neutrino fluxes two ways, either selecting random decay angles (i.e. unweighted Monte Carlo), or calculating the weight for that decay to produce a neutrino in the detector acceptance, a method developed by Rick Milburn of

Tufts University.

GNuMI, written by James Hylen and Adam Para at Fermilab[2], generates neutrino fluxes in a manner similar to PBEAM. It differs from PBEAM in being GEANT based, and in the larger number of effects that it includes. GNuMI was developed specifically for NuMI beam design. It includes code to properly handle the effect of polarization in the $\pi \to \mu \to \nu$ decay chain, including the angle and energy correlations, which is not part of GEANT.

	NUADA	PBEAM	GNuMI
Typical run time	0.2 hr	2 hr	200 hr
$\pi^{\pm}, K^{\pm} \to \nu_{\mu}, \overline{\nu}_{\mu}$	yes	yes	yes
$K_L^0 \to \nu_{\mu}, \overline{\nu}_{\mu}, \nu_e, \overline{\nu}_e$	no	yes	yes
$\mu^{\pm} \to \nu_{\mu}, \overline{\nu}_{\mu}, \nu_e, \overline{\nu}_e$	no	yes (ignores polarization)	yes
3 body decay model	none	phase space	V-A
Hadron absorption by horns etc.	yes	yes	yes
Secondary interactions from horns etc.	no	no	yes
μ (for monitor chambers)	no	yes	yes
Baryons (monitor chambers, radiation)	no	no	yes
Unweighted decays	no	yes	yes
Weighted decay to detector	K, π	K, π	K, π, μ

Table 9.1: Comparison of programs used for neutrino beam simulation.

The speed of NUADA is useful when a large number of variations of parameters are to be considered, but care must be used when interpreting the results. The wide band beam horn shapes were optimized with NUADA. The alignment studies used PBEAM's more realistic Monte Carlo tracking, at some cost in speed. GNuMI's larger range of physics effects are necessary for background studies of wrong-flavor neutrinos, and for calculating effects of secondary production from the horns and decay pipe walls. Table 9.2 shows the list of decays which contribute significantly to neutrino production in NuMI, and how they are modeled in GNuMI.

The work necessary for beam simulation and comparison with data will probably involve:

- Replacing the current GEANT/FLUKA model of hadron interactions in the target with another model; perhaps with an updated version of FLUKA or MARS or with data from a dedicated measurement of the production spectra using the NuMI beam and target.

- Developing techniques to use simulated events in the near detector in a weighted fashion, or to use data driven event reconstruction efficiencies, since brute force simulation of events to model the time, position, and energy beam dependence in the near detector would be too expensive in CPU time.

- Making multiple runs of GNuMI or GNuMI-like beam simulation, with variations of production and alignment parameters.

Parent	$c\tau$	Daughter	Branching Ratio	Type
π^+	7.80 m	$\mu^+\nu_\mu$	100 %	Isotropic
π^-	7.80 m	$\mu^-\nu_{\overline{\mu}}$	100 %	Isotropic
K^+	3.71 m	$\mu^+\nu_\mu$	63.51 %	Isotropic
		$e^+\nu_e\pi^0$	4.82 %	Isotropic V-A
		$\mu^+\nu_\mu\pi^0$	3.18 %	Isotropic V-A
K^-	3.71 m	$\mu^-\nu_{\overline{\mu}}$	63.51 %	Isotropic
		$e^-\nu_{\overline{e}}\pi^0$	4.82 %	Isotropic V-A
		$\mu^-\nu_{\overline{\mu}}\pi^0$	3.18 %	Isotropic V-A
K_l^0	15.49 m	$\pi^-e^+\nu_e$	19.35 %	Isotropic V-A
		$\pi^+e^-\nu_{\overline{e}}$	19.35 %	Isotropic V-A
		$\pi^-\mu^+\nu_\mu$	13.50 %	Isotropic V-A
		$\pi^+\mu^-\nu_{\overline{\mu}}$	13.50 %	Isotropic V-A
μ^+	658.65	$e^+\nu_e\nu_{\overline{\mu}}$	100%	Polarized V-A
μ^-	658.65	$e^-\nu_{\overline{e}}\nu_\mu$	100%	Polarized V-A

Table 9.2: Decays which produce neutrinos in `GNuMI`.

Based on experience with the current version of `GNuMI`, approximately one CPU-year will be required for the beam simulation, in addition to the time required for simulation of the events in the near detector.

9.2.2 Detector event simulation

The simulation of neutrino interactions in the detector is a significant portion of the MINOS computing effort. In order to accomplish this task, a GEANT-based Monte Carlo program `gminos` has been written. The `gminos` program combines a flexible description of the detector geometry, the flux from `GNuMI`, our best understanding of the neutrino interaction physics, and the simulation of the properties of the scintillator and photodetectors with the standard GEANT-supplied tracking and particle interaction routines.

Runs of `gminos` are controlled by FFREAD data cards which describe the run parameters; the geometry configuration; the event generator switches; and the tunable parameters in the active detector response. A `gminos` output file is an ADAMO structured file containing the data models and the actual data for once-per-run information (such as the geometry) followed by individual event records. Figure 9.1 shows a block diagram of `gminos`.

Most MINOS collaborators eschew the actual running of the `gminos` program which is generally left to a few experts. Conditions for runs are agreed upon by the collaboration as a whole. The experts set up the data card files to match the conditions and submit the jobs to various machines (including a farm of batch nodes) for event generation. At the time of this report the collaboration has generated hundreds of thousands of MINOS neutrino interactions under a variety of conditions. These files are available from a central location on the Fermilab AFS file system. Individuals with particular needs for runs with special

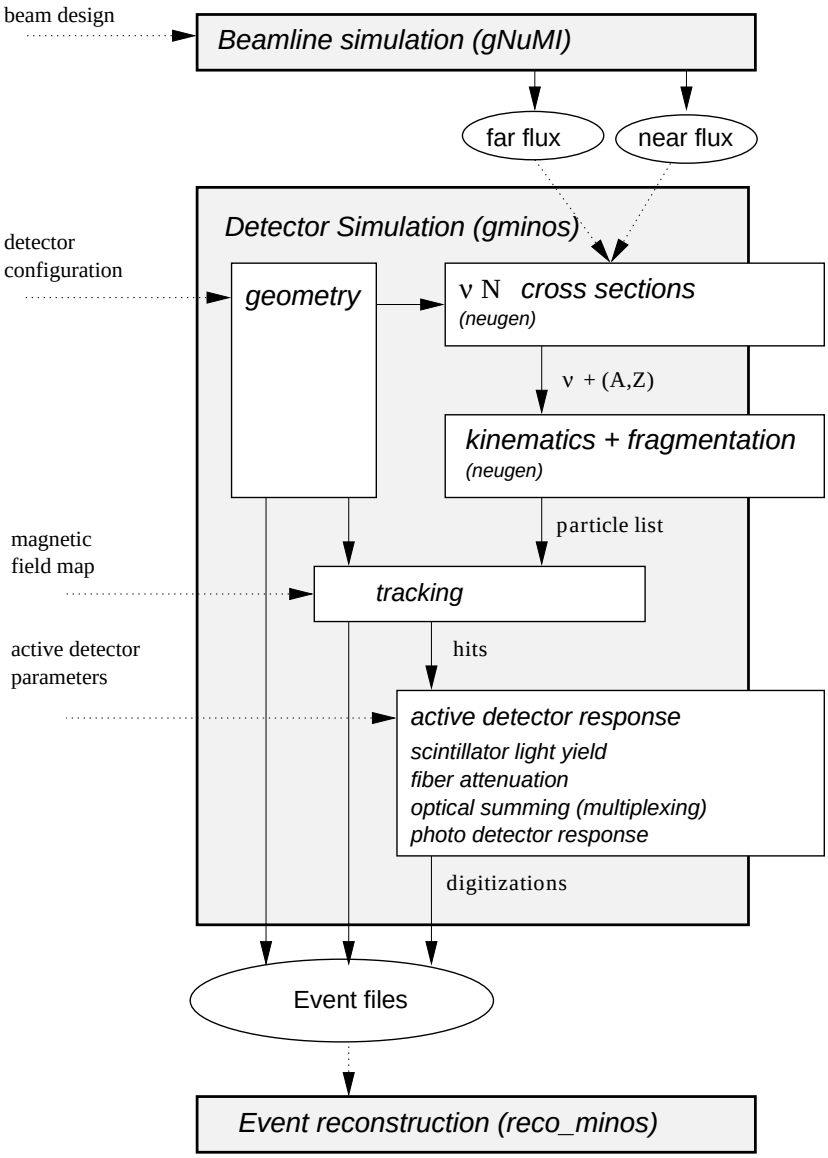

Figure 9.1: Block diagram of the `gminos` program. The major units of the detector simulation are shown. Their relationships with each other and outside elements are diagrammed schematically.

conditions are encouraged to run gminos on their own with support, as necessary, from the experts.

Modularity of the code functionality has been stressed so that sections of the code can be replaced without undue adverse effects. The following Sections describe the major components to the gminos program.

9.2.2.1 Interface to GNuMI flux

The output of the GNuMI simulation of the beamline is condensed into intermediate flux files by a separate stand-alone program. The flux files are a convenient format for use by gminos. By using a standard file format for the interchange, a user can change the flux used by gminos by simply changing a data card and supplying the new flux file. This provides a means for exploring the effects of different beamline configurations. Efficient near versus far detector event generation can be achieved by using different flux files derived from the same original GNuMI beamline simulation, resampled over the appropriate solid angle subtended by the detector.

In the case of the far detector, events can also be generated using a wide, artificial beam spectrum which can be weighted to simulate any of the three PH2 beam configurations (high, medium, low energy). This is possible because the beam at this location has a negligible divergence so there is no need to worry about the correlations of the neutrino direction with neutrino type and energy. The weighting factors are derived from histograms of the GNuMI flux. This approach reduces the need for a large Monte Carlo data sample for each beam configuration, which saves both disk space and CPU processing time.

9.2.2.2 NEUGEN: The MINOS event generator

All gminos simulations use the NEUGEN neutrino event generator to model the neutrino interaction, producing from an input neutrino and nucleus type a list of final state particles which are then returned to the detector simulation. In this Section we present a brief overview of the physics of the event generator and plans for further improvements.

At low energies, charged current neutrino interactions are predominantly quasi-elastic and single pion production, in which the neutrino scatters off an entire nucleon rather than the constituent partons. The cross section for quasi-elastic scattering is expressed in terms of the weak form factors of the nucleon. The vector components can be related to the well-measured electromagnetic form factors via the CVC hypothesis, and the axial vector form factor has been measured in numerous low energy bubble chamber (100 MeV - 10 GeV) experiments. For tau production, retaining terms proportional to the produced lepton mass leads to a significant contribution to the cross section from the pseudoscalar form factor. The contribution from this form factor is negligible for muon or electron neutrino scattering, and as such it is at present unmeasured in neutrino interactions. For the purposes of our simulations a theoretical expectation based on the PCAC hypothesis is used. Generation of single pion final states through resonance production is based on the neutrino production model of Rein and Seghal[5] and the Feynman, Kislinger and Ravndal model of baryon resonances[6]. This model treats the baryon resonances as the excited states of the 3-quark system bound by a relativistic harmonic oscillator potential. The matrix elements for neutrino induced res-

onance production are then calculated directly from the bound state wavefunctions. Single pion production is dominated by production of the $\Delta(1232)$.

At higher energies, the neutrino scatters off the partons within the nucleon. Neutrino deep inelastic scattering has been studied with high precision in the energy range 10 to 200 GeV by a number of experiments. DIS cross sections are again written in terms of form factors which can be now be expressed in terms of the constituent parton distributions. Tau production introduces an added complication in that form factors which are usually negligible (W_4 and W_5) must now be retained. Although unmeasured, their expectation in terms of parton distributions is known. Hadronization of DIS-generated final states is done using a scheme based on KNO scaling. The KNO model used has been tested and shown to be valid for neutrino induced hadronic final states.

NEUGEN also needs to take into account the fact that the nucleons participating in the interactions are not free but are bound within the nucleus. The two most important nuclear effects for low energy scattering are the Fermi motion of the struck nucleon and Pauli blocking of interactions with small momentum transfers. Both effects are modeled with a Fermi gas model of the nucleus. In this model all nucleon energy levels up to the Fermi momentum P_f are considered to be filled (thus generating the Fermi momentum spectrum), and momentum transfers which leave the final state nucleon with momentum smaller than the Fermi momentum are not allowed.

NEUGEN grew out of the neutrino event generator used by the Soudan 2 collaboration to model atmospheric neutrino interactions[7]. In this capacity the generator has been in use since 1987, and has been well tested - particularly at the lower neutrino energies of atmospheric neutrinos. Numerous comparisons to published data have been made, and experimental DSTs from the BEBC experiments, which took nearly 750,000 bubble chamber pictures in runs from 1977-1983, have been made available to the collaboration for more detailed comparisons which are currently underway. NEUGEN has also been made available to other neutrino experiments.

Ultimately one would like to address the question of the extent to which uncertainties in the physics models will affect the sensitivity of a given experiment. As a two-station experiment, MINOS is to first order insensitive to such uncertainties, as has been shown in previous studies. Nevertheless, it is for such studies that one would like to have a generator which incorporates the full range of physics models which have been proposed. One then uses existing data to determine the models and ranges of model parameters which are consistent with current measurements. This process of model inclusion and data comparison is an ongoing one which will continue over the next few years as NEUGEN continues to evolve and improve.

9.2.2.3 Interface with NEUGEN

NEUGEN outputs a list of particles in STDHEP form. The `gminos`-specific code pulls a neutrino from the flux file; samples the detector along the neutrino's path; decides on whether an interaction occurs; chooses a vertex position and nucleus type; calls the kinematics generator and enters the STDHEP list into GEANT's list of particles to track. This procedure correctly accounts for the distribution of material along the neutrino path and the relative proportions of nuclei, based on the geometry of the current run. The code is modular enough

that the cross section or kinematics routines can be improved or even completely replaced without major impact on the `gminos` code downstream from the interface.

9.2.2.4 Geometry

Although the finest-level details of the MINOS geometry are still being defined, handling future changes is not expected to require significant new code. The `gminos` code provides a simple user interface which allows substantial reconfiguration of the detector for a wide variety of parameters. These parameters are then converted into standard GEANT geometry descriptions of the detector. By making the geometry description sufficiently abstract one can describe both the near and far detectors without need for separate parallel code.

9.2.2.5 Tracking and hit storage

Particles are tracked through the GEANT geometry in the usual manner. Different maps for the magnetic field can be set along with the geometry specification. These field maps are generated by the MINOS magnet group (Chapter 4) and are specified relative to a single steel plane's local coordinates. The `gminos` code then performs the appropriate coordinate transformations.

Pertinent information is recorded for each particle traversing an active detector volume. These attributes include the volume identifiers, energy deposition, entering and exiting positions. These objects are designated as *hits* and contain the exact, unknowable information about the particle's traversal.

9.2.2.6 Digitization

The final step of the detector simulation process is collecting together the *hits* in a volume and converting them into *digitizations*. The digitizations (digits) are the combined effect of individual particles interacting within the active volume. These digits will closely mimic the types of signals that come out of the front-end electronics, described in Chapter 6. This process of modeling the active detector response encompasses the light production (including Birk's Law saturation effects[8]) in the scintillator, light collection and re-emission in the wavelength shifting fiber, attenuation in the fiber and photodetector response. The final result gives realistic results for the photoelectron statistics.

We have chosen to retain the *hits* in the data files. This imposes a large space penalty but allows us to reuse the same events while varying the specifics of the digitization process. By doing so we can investigate the effects of different scintillator light yields, fiber attentuation, and other possible changes in instrumentation details. The redigitization can be done in the analysis framework (described in Section 9.2.3) just prior to the event reconstruction.

9.2.3 Event reconstruction

A framework for event file processing has been developed: the `reco_minos` package standardizes the reading and processing of files currently generated by the `gminos` program and, when the time comes, real events written in a compatible format. This shell incorporates

ADAMO's ability to skip event input based on information in the header bank, allowing quick access to selected events in a file. Figure 9.2 shows a flow diagram of `reco_minos`.

At a minimum, the user need only supply routines for their own histogram booking and event processing, a list of event files and a set of data cards for controlling the analysis. Hooks are provided to allow users to supply routines to handle different phases of file processing.

The user's event processing routine can call upon collaboration supplied routines for basic reconstruction. These routines are still being developed and refined. While it is likely that much of the currently available code will not survive unmodified to the time data-taking begins, it is has been a good introduction to what the final requirements will need to be. This insight will lead to better algorithms in the future. Even these incomplete algorithms have provided feedback for use in detector hardware decisions.

9.2.3.1 Demultiplexing and attenuation correction

A framework for simulating the optical summing (multiplexing) of multiple fibers on single photodetector pixel has been developed. Simultaneously, a program to disentangle the multiplexing is being tested.

An algorithm has been written to account and correct for the light loss due to the attenuation in the wavelength shifting fiber, using nearby strips in the orthogonal view to determine the average position in a cell. This correction is necessary for achieving the good energy resolution which is intrinsic to the MINOS scintillator detector technology.

9.2.3.2 Vertex finding and event separation

A generic vertex finder has been written and gives adequate results. Alternative finders that improve on this for specific event topologies can be added to run in parallel.

Algorithms for separating out simultaneous events in the detector have not yet been considered.

9.2.3.3 Muon reconstruction

The problem of tracking and fitting muons in MINOS is more complicated than in conventional detectors as they lose a large amount of energy in traversing the steel plates, many coming to rest. Furthermore the toroidal field is different in direction at each point along the track and multiple coulomb scattering plays a large role in defining the track trajectory. A iterative least squares method of coping with these difficulties, patterned after the CDHS approach[9, 10], has been adapted for MINOS. Muon reconstruction is performed in two stages.

First, the hits associated with the muons are found by searching for track segments in each view. The longest segment is taken as the track basis and extrapolated out to the edge of the detector (or end of the track) and back to the hadron shower. Currently it does not attempt to extrapolate back to the vertex in the shower but ultimately this should be possible. Second, the muon hits are fitted to a curved trajectory. By tracking the muon through the detector material, the energy loss of the muon and its multiple Coulomb scattering can be calculated, producing a full non-diagonal weight matrix for the χ^2. The trajectory equations can be then solved by iterative least squares, yielding values for the parameters plus a full

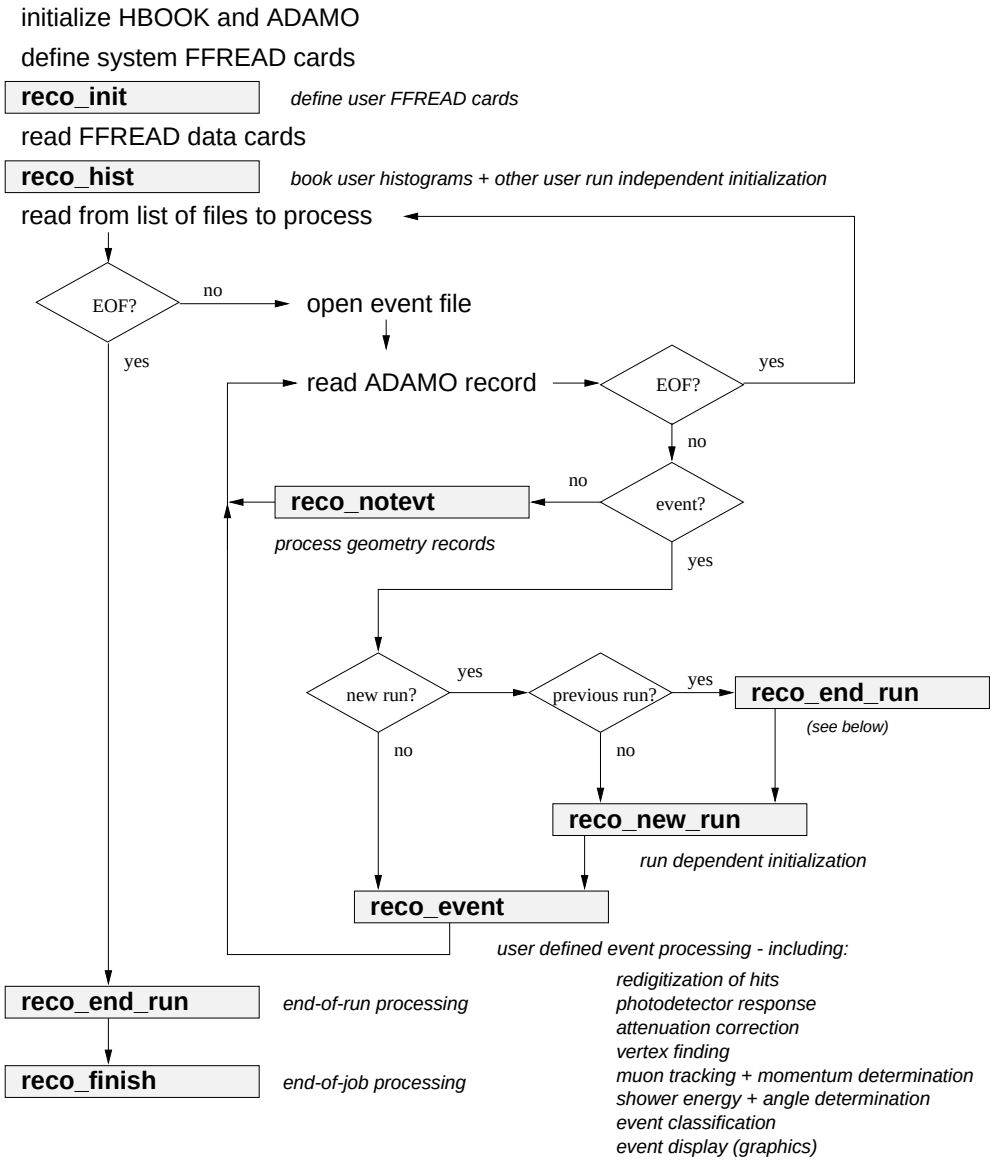

Figure 9.2: Flow diagram for the `reco_minos` program. The routines that users write are shown as shaded boxes. Dummy routines are used if no routine is supplied.

error matrix. However, when the muon is close to stopping, its increasing departure from linearity means that care must be taken to avoid the final segment of the track in the fit. This procedure yields momentum errors of $\sim 13\%$ and angular resolutions of ~ 0.03 radians when muons are fitted over ≤ 400 planes.

For short, stopping muon tracks that are fully contained in the detector, a simple track length measurement will result in an even better estimate of the muon energy, although curvature is still necessary for determining the charge.

9.2.3.4 Shower energy and angle determination

Preliminary routines have been written for energy (photoelectron summing) and shower angle determination, but have not been extensively tested. In particular the energy sum is dependent on the attenuation correction; energy resolutions significantly suffer if it is not applied. The existing code does not yet attempt to remove the overlapping muon track, but this will be remedied in the future as both this code and the muon tracking code improve.

9.2.3.5 Event identification

We have used event classification heuristics (NC versus CC, ν_e versus ν_μ versus ν_τ, etc.) in various studies of MINOS sensitivity to neutrino oscillations[2, 11, 12]. We have not mounted a systematic attempt to generate the best-possible approach for all cases. Rather, estimates of the signal efficiency and background rejection set a lower limit on how well the experiment might perform. More sophisticated approaches will give some improvement but are not expected to change the conclusions of our current analyses.

The NC versus CC classification proceeds on two levels. The simpler approach is to not attempt event-by-event classification. Instead, we separate the events into two classes that to a large degree overlap the physical process, but are easily identified in real event topologies where an estimate of the cross contamination can be made. This is the approach of using "short" and "long" events as initial estimators of the NC and CC events. Once so classified then the unfolding of these into NC and CC events proceeds at the statistical level. The second approach to NC versus CC classification is to attempt pattern recognition of the muon track. This is part of the muon reconstruction code described above.

We have developed techniques for neutrino flavor determination in two broad categories. One approach has been to use cuts on the distributions of event characteristics and reconstructed quantities to distinguish the categories. The other methodology uses essentially the same quantities but presents them to an artificial neural network (ANN) for identification. The ANN is first trained to classify the events by presenting it with a test sample of known types. It is then tested with a separate sample of events; the result of the test is a value between 0 and 1 that serves as an estimator of how likely it is to be of the type represented by the output value. As the threshold on this output value is increased, the efficiency for correctly classifying the signal goes down, but the rejection of background goes up. The optimal value for the threshold may depend on the exact analysis in which the classification is being used. Generally the cuts and ANN approaches have yielded very comparable results in tests where both have been attempted. It is expected that the ANN method should generally lead to a slightly better separation.

Explicit identification of electrons is based on the fact that electron showers are shorter, narrower and denser than hadron showers. Thus, after selecting "short" events, cuts (or ANN inputs) based on the charge distribution in the remaining shower produce ν_μ charged current rejection factors of 700 for ν_e's. The charged current efficiency of 14% is sufficient to give limits for $\sin^2(2\theta)$ of 2×10^{-3}. This study was performed using the high energy beam configuration, but we have also developed similar algorithms for the low and medium energy neutrino beams.

Identification of explicit τ production can be made in various τ decay modes. In $\tau \to \mu\nu\nu$ and $\tau \to e\nu\nu$ the analysis is based on selecting quasi-elastic (low-y) τ production, where there is little hadronic activity at the production vertex. Then, for a given beam ν energy, the kinematics of the events with missing ν's gives lower lepton energies than for equivalent ν_μ events. Thus in the narrow band beam, kinematically unambiguous τ production can be observed, provided the tails of the beam energy distributions can be kept under control. The decay $\tau \to \pi + X$ can be isolated by selection of high energy π's which interact to produce hadronic "stars". The background from neutral current π production is suppressed because the energy distribution of these π's is much softer than that from τ decay. Limits on $\sin^2(2\theta)$ down to around 0.2 can be obtained[13].

All of these explicit tests for ν_e and ν_τ production involve detailed reconstruction of the hadron shower, including possibly the reconstruction of individual tracks and showers within the overall hadron shower. Techniques for this reconstruction are still rudimentary but are expected to be significantly improved before data is available.

9.2.3.6 Graphics

Computer graphics plays an important role in many aspects of an experiment. For the standard tasks of data analysis, fitting, and presentation graphics MINOS currently uses CERN software, including PAW and HIGZ.

Graphics are also used to display simulated data and reconstructed events both for algorithm development and to demonstrate event types and topologies. An X-based, interactive 3D graphics system based on VINES (Erik Gottschalk, Univ. of Illinois) is available on a few of the computer platforms used by the collaboration. A limited, static event display using HIGZ and HPLOT routines is available on all platforms. Rudimentary information about hit location and pulse height are displayed, but without any interactivity. Users can supply calls to familiar routines to add additional information.

We have begun work to produce a more flexible and portable display based on an OO paradigm. Two efforts are currently being pursued. In one project the MINOS event data and analysis code are being integrated into the ROOT software system currently under development at CERN by Rene Brun and collaborators. In the other, we are investigating the use of OO-based Internet tools to produce portable events displays. A prototype display employs Java and VRML to produce 3D virtual-reality event displays and associated analysis tools.

9.2.4 Code management

9.2.4.1 Code manager/librarian

The responsibility for managing packages in the MINOS collaboration central software repository and for maintaining their functionality will be vested primarily in the designated librarian or code manager. Deputy managers may have responsibilities concerning details of individual packages/libraries. In particular, the code manager will be responsible for:

- promulgating and enforcing coding practice standards
- regulating changes to the central repository
- informing collaborators of the use and availability of outside software packages required by collaboration code.
- disseminating documentation of collaboration code (via the World Wide Web).
- maintaining E-mail distribution lists for current software news and discussion.

9.2.4.2 MINOS software repository

The central repository makes the common code accessible to all members via the program CVS (Concurrent Versions System[14]). CVS allows access to a central repository by a remote computer. Thus it is only necessary to have a complete repository at one location, helping to minimize the possibility of divergence in collaboration-standard programs. The MINOS source code repository will eventually grow to include all collaboration-standard programs for simulations, event reconstruction, physics analysis, and event display. The GEANT-based detector simulation `gminos`, as well as analysis/event reconstruction routines in the context of the `reco_minos` program, are now available in the CVS repository. Checking a program module out of the repository provides the user with copies of all the source code files as well as the `Makefiles` necessary to construct libraries and executable programs.

9.2.4.3 Software development and distribution using CVS

CVS is a tool for version management and code distribution during the software development phase. Built on top of the older RCS (Revision Control System), CVS adds the flexibility of allowing multiple developers to work on the same source files concurrently. Each programmer/user checks out copies of the desired files from the central repository into a personal work area on his/her home computer. The local copies of the files can be modified or deleted as the user desires without effect on the standard repository version. The repository version is only changed upon the explicit request of the CVS user. Thus the system accommodates both users who wish to modify the source code for their own use or for redistribution to the collaboration, as well as those who merely want to build a standard executable program for their studies.

Once a file is stored in the CVS repository, a complete history of its evolution thereafter is recorded by CVS. Any version of a file can be reconstructed at any time based on one of the identifying characteristics: date, revision number or symbolic tag.

CVS has been designed to help resolve the inevitable conflicts that arise when more than one person edits a particular source file. CVS does not use file locking which would prevent concurrent development. Rather, CVS has conflict resolution algorithms which sense any incompatibility between the changes to the local copy and a new repository version. Changes that can not be resolved generate warnings and in-file delimited code lines. The user must then decide whether to alter their code to incorporate the newer revision, or to confer with the colleague who committed that revision.

The central repository can in principle be updated at will by a user with the changes they have made to their local source files. This introduces a conflict between program developers who want the most up-to-date version of the software and are prepared to tolerate bugs, and users who want only functional, reasonably well-tested code. To resolve this, we are developing a set of regulations concerning who may actually commit code and requirements on the functionality at the time it is committed. To date this has been generally very informal, but as the amount of code and number of users increase so will the level of software management with alpha, beta, and production code releases controlled by individual package managers.

9.2.5 Requirements to complete the Fortran offline system

Although a considerable amount of code has been written, there still remains a lot of work to complete a user-friendly, well-engineered system. The code will need revision and reworking to improve and extend its functionality. The present system also needs the addition of a database for storage of run and calibration constants along with all the miscellaneous data that are needed, as well as the raw detector data. The current graphics package is very rudimentary and will need complete reworking to be made into a flexible tool for diagnostic and debugging work.

It is difficult to estimate how much effort would be required to produce a system adequate for data taking in 2002, but it probably exceeds ten man-years. We believe that this effort can be found within the MINOS collaboration. The purchase of commercial software will probably not be required except for the (Oracle) database system.

9.3 The OO alternative

9.3.1 Motivation

It is difficult to overstate the importance of computing in experimental HEP. It is central, and essential, to all phases in the life cycle of an experiment, from detector design and construction through to its operation and analysis of the data it produces. Without the dramatic advances in computing technology over the past 30 years most of the HEP program would have been impossible. The clear lesson of this is that we have to keep up to date with mainstream developments if we are to continue to exploit these advances.

The disparity in the rate at which hardware performance improves compared to that of software has long been recognized. While hardware performance growth is essentially exponential, software growth is almost linear. A major part of this disparity comes from the way

hardware engineers manage complexity. By structuring systems into subsystems and hiding complexity behind simple interfaces, development is simplified. Individual subsystems can evolve independently and still remain compatible, while new systems can be produced by assembling the subsystems in different configurations. OO is the latest, and most successful, in a series software engineering paradigms that have attempted to achieve the same level of structuring in software. Modularization is achieved by hiding complexity through encapsulation. Larger systems are broken down into more fundamental ones by abstraction. These subsystems can then evolve over time through the mechanism of inheritance, while assemblies of evolved subsystems continue to cooperate.

OO is relevant to experimental HEP because of the common problem domain all experiments address. Common solutions to some of these problems have been very successful, for example ADAMO, which has been described above. Its success comes from its OO-like properties. Like any well defined class, it presents a simple rugged interface to the user and provides access mechanisms to the data it holds. However, it does not hide its data and is of limited help in building modular systems. Until now, most standard HEP software systems have provided support for the major off-line activities of Monte Carlo, Data Reduction, Event Reconstruction and Analysis, but have stopped short of providing these functions themselves. So each experiment has had to devote many man-years to developing code to do this, despite the fact that there are many shared problems and solutions. To pick just one example in MINOS, we have to be able to find and fit muons in our data and are producing code to to this, despite the fact that such codes were analyzing bubble chamber film 20 years ago. The code robustness that is inherent in the OO model is particularly important in HEP, where people, with a wide range of skills and understanding of the code, have to work with it. A properly designed system should allow much greater access to the code to those outside the core support group, because it is more modular, with fewer side effects to catch the unwary! Its stricter disciplines ensure cleaner code and facilitate code development, a process that continues throughout the course of an experiment.

As has been stated earlier, the offline software requirements of MINOS are very modest; it is not the case that OO is essential. Indeed, in the short term, an OO alternative will involve more effort. In the longer term this should be repaid in the reduced maintenance compared to systems such as ZEBRA and ADAMO where support is already becoming very fragmented. The simplicity of this experiment also makes it a very good one in which to migrate to OO, a migration that is already underway in all of the larger, next generation experiments. Beyond the fact that it will give us a natural interface to GEANT4, the OO replacement of GEANT3, it will ensure that we stay in the mainstream of HEP code development. Predicting the style of computing 5 or 10 years from now cannot be done with any great certainty beyond stating that it will be much more powerful and will be different! By investing the additional effort to master OO now we will maximize our ability to best exploit improvements in the technology.

A small group within MINOS is in the process of studying all the consequences of choosing OO as the basis of our off-line software. The remainder of this Section has to attempt to second guess what that group will learn.

9.3.2 Requirements for an OO alternative

Choosing the OO route would lead to a number of extra requirements that would not be necessary for the Fortran route. At least initially there will inevitably be extra costs and effort required. However neither is expected to be large on the scale of the full experiment.

9.3.2.1 Training

Members of our collaboration have little experience with OO and will have to develop their own set of experts. Fully understanding the subtleties of a language like C++, and how properly to analyze and design an OO system, demands a higher level of expertise than that required to understand Fortran-77 and to perform procedural analysis and design on a system based on a data structure manager. The conventional wisdom is that it takes of order 6 months to turn a good Fortran programmer into a good OO programmer. Although the same level of expertise is not required of the rest of the collaboration, anyone who wants to make a significant contribution to the software will have to become a competent C++ programmer which could take of the order of a month. New post-graduates joining will probably already be familiar with C++. We estimate that a total of about three man-years of training will be required within the collaboration.

9.3.2.2 CASE tools

The relationships between classes in OO are much richer than the relationships between routines of a procedurally based code. Classes may inherit, own or simply use other classes in a variety of ways. This richness makes the use of CASE (Computer Aided Software Engineering) tools much more compelling, to help people express, exchange and check consistency of the systems they are studying. If MINOS uses OO then we will probably need to buy some CASE tools although the number of licenses can be restricted to core group designing the heart of the system. Typical costs for popular tools would be around $10,000. However we do not currently know what tools we need, and might even decide to do without any.

9.3.2.3 Commercial packages

The standardization of interfaces between classes promotes the ability to integrate software from different sources. At least some parts of the HEP community are now concentrating their programming efforts on problems that are unique to the discipline and seeking commercial software solutions to more general problems. Indeed this is the philosophy behind LHC++: it puts great emphasis on the use of commodity software. They plan to use only one major software component entirely developed within HEP, namely GEANT4, an OO version of the GEANT simulation package.

This of course leads to another cost for any experiment that takes this approach. The whole collaboration will need to run the software; many more licenses are required than for CASE tools. Estimating the cost of this approach is fraught with difficulty: the result depends crucially on our computing model, something that the MINOS OO working group is studying. At one end of the spectrum of choice lies the LHC++ model, which will be expensive. At the other end is the tradition of sharing software written by and for the HEP

community. Here the only current candidate is ROOT, a system being developed at CERN for NA49 and also being studied by a number of other experiments. It appears that we could use both GEANT4 and ROOT as the basis of our computing model without the need for any commercial software. There are several advantages to this approach:

- Cost. This could be a zero cost option, at least in financial terms.
- An optimal match to the requirements of HEP. There is no need to support generic requirements that are not part of the HEP problem domain.
- Better control over its development. It responds directly to the changing needs of HEP.
- Long term security. It belongs to HEP with its own unique development time scales.

The last of this list, long term security, is currently the weakness with ROOT. Until further experiments invest in it, its long term future is uncertain.

9.3.2.4 An OO version of the offline system

The MINOS software group is now devoting significant effort to developing a new OO equivalent of the Fortran based system described above. Others working in OO have told us that the only way to learn is by trying and that it is very hard to get it right the first time. Here MINOS has an advantage, relative to other experiments embarked on this course, in that its requirements are comparatively simple. So we could plan to develop a first version on a one year time frame and then use the experience gained to build a second on a similar time frame. We estimate that this would require about 10 man-years of effort in addition to the effort involved in the updating and improving the algorithms already developed. As for the Fortran alternative, we expect that most of this effort will be available from the collaborating groups.

9.4 CPU and storage requirements

In this Section we attempt to estimate the amount of cpu power and data storage that will be required to process and store the data from the MINOS far and near detectors, as described in Chapters 5 and 6, and to perform high statistics Monte Carlo simulations of the experiment.

Such estimates are notoriously subject to under-estimate, particularly when the program systems are not complete and real data has not been experienced. We have tried to make realistic estimates, based on currently working code, including an allowance for reprocessing of data. However even with existing code, the variation in timing across different platforms is large. Also very little effort has currently gone into optimization of code and large factors may well be available. With these uncertainties the estimates given here are probably not accurate to better than a factor of two. Even thus inflated the requirements of MINOS are modest by today's standards. With the natural progression of computer power we expect to be able to keep pace with any unanticipated increase in the computer requirements or reduce our hardware requirements. However, if the worst should occur and both these estimates

are over-optimistic and extra computer power is not available, it should be noted that in the estimates below the vast majority of time is spent processing secondary data; cosmic ray muons or neutrino events outside the beam spot in the near detector. Fast filters for real physics data can be developed and the remainder of the data either sampled or processed with faster, less complete, algorithms.

The timings are in terms of a modern 300 MHz RISC processor which is equivalent to around 200 MIPS.

9.4.1 Far detector

We assume:

1. A trigger rate of 2 Hz, made up of 1 Hz of cosmic ray muons and 1 Hz of background noise (radioactivity or electronics). The 22,000 neutrino interactions per year are negligible in this calculation

2. An average of 300 hit scintillator strips per trigger. This is probably generous because noise events will be small. There are 192 scintillator strips in a plane. Most cosmic muons will hit fewer strips than this but high energy muons have a high probability of producing a showering bremsstrahlung electron in their passage through the detector.

3. Eight bytes per hit read out by the electronics

4. Five seconds processing time per event for a cosmic ray muon. Noise triggers will be fast, therefore the average processing time per trigger is 2.5 seconds.

5. Data expansion during processing by a factor of 5.

6. Continuous far detector operation with a 90% duty cycle. Running outside beam-on periods is probably necessary, first to obtain sufficient cosmic ray muons for calibration and second to be continuously sensitive for atmospheric neutrinos and other cosmic ray phenomena.

7. Storage of raw data and processed information for all cosmic ray muon events. This will probably not be necessary, after an initial running-in period. The muons are mostly required for calibration purposes and the data for this will be filtered off and stored as histograms for each scintillator strip. However, if raw muon data are not kept, the storage requirements are negligible so this represents a worse case.

8. At least in the first stages of data taking it will be necessary to reprocess data as the reconstruction algorithms are refined and developed. Since continuous operation is assumed, this will require that we double the computing power available to enable us to reprocess in parallel with data taking.

Table 9.3 gives some of the far detector requirements which result from this model.

It can be seen from the Table that a modest farm of 5 cpu's will be adequate to fully reconstruct all the cosmic ray muons, even assuming no increase in speed beyond today's models, and assuming that this process is necessary for calibration. Allowing for the data

Triggers/year	$2 \times 3 \times 10^7$	6×10^7 triggers
CPU processing time/year	$2.5 \times 6 \times 10^7$	1.5×10^8 sec
Readout bytes/year	$8 \times 300 \times 6 \times 10^7$	1.44×10^{11} bytes
Data stored/year	$5 \times 1.44 \times 10^{11}$	720 Gigabytes

Table 9.3: Summary of estimated far detector cpu and storage requirements, including generous contingency allowances.

reprocessing we will require a farm of 10 processors at the Soudan site. The data storage requirement is small by today's standards, even if we were to keep all the cosmic muon raw data.

9.4.2 Near detector

Rates in the near detector are much higher than in the far detector and the processing and storage requirements are more stringent. We assume:

1. Thirty ν events/spill in the target and veto regions of the near detector. Of these, the 0.5 events/spill produced in the central 25 cm radius of the target region are required for physics comparison with the far detector. The remainder will be used to monitor and model the beam and for other nonoscillation physics that may be performed with the near detector.

2. Event rates of 15 muons/spill entering the target region from upstream neutrino interactions and 270 Hz of cosmic ray muons crossing the full detector, in addition to the neutrino interactions. For calibration purposes the upstream muons, and an equal sample of the cosmic muons, will be fully reconstructed for each spill. Full reconstruction of the remaining cosmic muons is probably unacceptable for cpu time reasons but they may be used in a simple histogramming mode to obtain very high statistics calibration data. It may be that ultimately the full reconstruction can be dispensed with.

3. A negligible in-spill random trigger rate.

4. An average of 100 hits/trigger. Near detector events are smaller than the cosmic ray muon events in the far detector, the muon spectrometer sampling is coarser and the lower energy muons will not produce large bremsstrahlung showers.

5. Eight readout bytes/hit and 5 times the raw data stored per trigger.

6. One second processing time per trigger.

7. An effective year of 10^7 seconds.

8. One complete reprocessing of the data.

Triggers/year	$60/1.8 \times 10^7$	3.3×10^8 triggers
CPU processing time/year	$1.0 \times 3.3 \times 10^8$	3.3×10^8 sec
Readout bytes/year	$8 \times 100 \times 3.3 \times 10^8$	2.6×10^{11} bytes
Data stored/year	$5 \times 2.6 \times 10^{11}$	1.3 Terabytes

Table 9.4: Summary of estimated near detector cpu and storage requirements.

Table 9.4 gives the numbers for the near detector quantities.

Ten processors running continuously will keep pace with the incoming data integrated over a full year. Allowing for reprocessing we will need access to a farm of 20 processors. This is an upper limit on the cpu usage since less than 1% of the events being reconstructed are required for MINOS neutrino oscillation physics. If there should be cpu limitations, more stringent cuts on the events reconstructed could be applied. Similarly, although the data storage requirement is not large, it may be much reduced if only the calibration information for each scintillator strip is stored.

9.4.3 Monte Carlo simulation

We expect 22,000 neutrino events per year in the far detector and 2×10^6 per year in the restricted target volume of the near detector. The Monte Carlo calculation has to:

1. Determine and correct efficiencies and biases in the reconstruction and selection processes in the far detector. The Monte Carlo statistics must be overwhelming compared to the number of events in the far detector, by at least a factor of 10.

2. Translate the distributions measured in the near detector to those expected in the far detector, making allowances for differences in the beam and the detectors. The requirement on the accuracy of this transformation is only that the statistical error should be negligible compared to the statistical accuracy of the *far* detector data. Thus a Monte Carlo sample equal to the near detector data sample will be adequate.

We thus expect to require a Monte Carlo sample of approximately 2×10^6 events per year for the combined near and far detector analysis. In order to study the algorithms used in distinguishing neutrino types based on event topologies, we must generate events for the three possible modes of no oscillations and and for oscillations to each of the other flavors, tripling the the number of events necessary.

The current cpu time required for generating an event in `gminos` is ~ 20 seconds. Adding time to generate the beam neutrino and for reconstruction we estimate a total cpu time of 40 seconds per event. The average storage requirements for Monte Carlo events will be substantially larger than data events. Two contributions to this increase are the additional space necessary for storing generated truth information and the desire to store intermediate *hit* information as well as the final *digitizations*. The hit information allows us to study the effects of uncertainty in light yield, photodetector gain variations and other detector effects without performing the cpu intensive generation and tracking of events. Based on the

empirical estimates from the current data sets, Monte Carlo events average about 57 kbytes per event. Table 9.5 gives an estimate of the total cpu and storage requirements for the Monte Carlo.

Events generated/year	$3 \times 2 \times 10^6$	6×10^6
CPU processing time/year	$40 \times 6 \times 10^6$	2.4×10^8 sec
MC event size		57 kbytes
Data stored/year	$57 \times 6 \times 10^6$	330 Gigabytes

Table 9.5: Summary of estimated Monte Carlo cpu and storage requirements.

A modest processor farm of nine machines dedicated to MINOS Monte Carlo event simulation will provide these events. Generating events under different conditions or regeneration to correct early deficiencies could again double these estimates.

9.4.4 Summary of cpu and storage requirements

We have shown that the data processing and storage requirements for the MINOS far detector and Monte Carlo data are quite modest, even if we fully analyze and store every trigger. The near detector data rates are much larger but even there the reconstruction and storage of the full event sample in the target region of the detector is well within the capacity of the present day facilities at Fermilab. We expect that, with the usual growth in capacity of the computer industry, by 2002 the load that MINOS places on computing facilities and the expense of providing them will be minimal.

9.5 Data processing model

Given the event rates and computing requirements described in Section 9.4, we construct the following model of the data processing and analysis for MINOS:

1. Far detector

   - The far detector data will be immediately reconstructed offline at the Soudan mine site. That is, events will be transferred, probably in run-size batches, from the DAQ system to a small processing farm where full reconstruction will be performed.
   - Candidate beam neutrino events (and other small selected data samples, e.g., candidate atmospheric neutrino events) will be filtered and written to permanent storage. They will be transferred to Fermilab to the central store, probably by Internet connection but possibly on a hard storage medium.
   - Calibration data from cosmic ray muons will be processed at Soudan and condensed to calibration data sets at the processing farm. The calibration sets will be sent to Fermilab for permanent storage and distribution to the collaboration.

There will probably be no requirement for raw cosmic ray data to leave the Soudan mine site.

The processing hardware requirements at the mine will be:

- A farm of approximately 10 cpu's, of a type to be determined by cost and performance in the year 2001.
- Disk storage sufficient for a few days data, around 50 Gigabytes.
- A permanent storage medium compatible with the Fermilab central data store in 2001.

2. Near detector

- The near detector data will be processed in a farm of around 20 processors at Fermilab. It is expected that these will be part of the Fermilab central processor farm. Instantaneous data rates during runs can be rather high, thus it may only be possible to process sample runs during data taking, with the remainder of the data written to permanent storage for processing during beam-off periods.
- Calibration data will again be condensed to calibration data sets and raw muon data will not need to be stored.
- Local disk storage will be required only for buffering, and data will be written directly to the Fermilab data store.

3. Monte Carlo

- Monte Carlo generation can be done on the Fermilab farm, or possibly at collaboration computer centers. The load is not expected to be large.
- Monte Carlo data will be stored centrally in the Fermilab data store.

4. Data distribution

- All physics neutrino events will be stored with raw data and processed quantities in the Fermilab data store. These will be accessible to all the collaboration via AFS or equivalent. If required, local copies of the raw data can be kept at collaboration computer centers. This is particularly likely to be the case for overseas collaborators where link speeds to the U.S. tend to be slow.
- Ntuples will be produced for physics analysis via PAW (or OO equivalent). These will be generated and stored centrally but are likely to be copied to local areas. Users may, of course, generate their own ntuples from the raw data.
- Calibration databases will be linked and up-to-date calibration data automatically distributed to local sites.

9.6 Summary

The MINOS computing requirements are not large. Stripped to the bare minimum, MINOS computing could be carried out on a handful of PCs. In practice, rather more than the minimum of calibration data and events in the near detector are likely to be processed and kept, at least at the start of data taking. Even with this extra data, the computing load from the near detector will be well within the capacity of the Fermilab processing farm and data store, so we propose to use these facilities for MINOS.

We plan to install some computing capacity at the far detector laboratory, if only to provide insurance against failure or lack of capacity in the links to Fermilab. However the total hardware requirement will be small, less than $100k for processors and storage.

The bones of an offline analysis system already exist, written under Fortran-77. Sufficient physicist effort exists within the collaboration to complete this system before data taking. The conversion of the system to an Object Oriented C++ form will require more effort in the short term, but may save effort in the long term, as support diminishes for the Fortran tools we use. A group of physicists and physicist-programmers are studying the OO possibilities for MINOS and, if the collaboration decides to take that route, this group will provide the majority of the effort. However, the addition of one or two programmers from the Fermilab computing department would greatly ease this process.

The collaboration will require a computer manager to take overall command of the computing system at Fermilab, and a second-in-command at the second site (presumably the Soudan mine). Also the collaboration needs a systems programmer, or systems oriented physicist-programmer, to take detailed control of the writing of the offline system. If these people cannot be found within the collaboration we will request them from the Fermilab computing department.

Chapter 9 References

[1] "Object Oriented Analysis and Design with Applications" 2nd edition, Grady Booch, Benjamin Collins 1993.

[2] The MINOS Collaboration, "MINOS Experiment R&D Plan: FY 1996-1998," June 1996, Fermilab report NuMI-L-184.

[3] WWW page http://www1.cern.ch/Adamo/ADAMO_ENTRY.html.

[4] D.C. Carey and V.A. White, "NUADA, the Fermilab Neutrino Flux Program", Note PM0011, Fermilab, June 1975.

[5] D. Rein and L. Seghal, Ann. Physics (N.Y.) **133**, 79 (1981).

[6] R.P. Feynman, M. Kislinger and F. Ravndal, Phys. Rev. **D3**, 2706 (1971).

[7] H. Gallagher and M. Goodman, "Neutrino Cross Sections," November 1995, Fermilab report NuMI-112.

[8] J.B Birks, "Theory and Practice of Scintillation Counting", Macmillan, New York (1964) p40.

[9] Muon Momentum Measurement in Magnetized Iron, A.Para, NuMI-L-222.

[10] Review of Track Fitting Methods in Counter Experiments, CERN Yellow Report 81-06.

[11] The MINOS Collaboration, "Status Report on τ Identification in MINOS," January 1997, Fermilab report NuMI-L-228.

[12] The MINOS Collaboration, "MINOS Progress report to the Fermilab PAC," October 1997, Fermilab report NuMI-L-300.

[13] D.A. Petyt, "$\tau \rightarrow \pi + X$ analysis in MINOS," October 1997, Fermilab report NuMI-L-258.

[14] "Version management with CVS, Release 0.9 for CVS 1.3+", Per Cederqvist, Signum Support AB, Linkoping, Sweden (1993).